1A4-01

なんでもRSS - HTML文書からのRSS自動生成

Automatic Generation of RSS Feed based on HTML Document Structure Analysis

南野 朋之[*1] 奥村 学[*2]
Tomoyuki Nanno Manabu Okumura

[*1]東京工業大学大学院総合理工学研究科
Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology

[*2]東京工業大学精密工学研究所
Precision and Intelligence Laboratory, Tokyo Institute of Technology

In this paper, we present a system to automatically generate RSS Feeds from HTML documents which include time-series information with date expressions (e.g., archives of weblogs, BBSs, chats, and mailing lists, update descriptions on a site page, announcements of events, and so on). Our system is based on extraction of date expressions, structure analysis of HTML documents, and title detection/generation from the contents.

1. はじめに

Web 上の情報は，人が目で見て理解する情報であったが，近年，計算機で直接扱うことができるメタデータを付加しようという動きが活発になってきている．ここ数年注目されているのが RSS[RSS-DEV Working Group 00, Libby 99, RSS at Harvard Law 02] (RDF Site Summary / Really Simple Syndication) であり，現在爆発的に広がっている Weblog では，ほぼ標準で RSS によるメタデータ配信が行われるなど，広く利用されるようになった．RSS は，サイトの更新情報 (見出し，要約，本文，更新日時など) を配信するのに適したフォーマットであるため，多くの新聞社の Web サイトやニュースサイトなどでも利用されており，今後も RSS を配信するサイトの数は増え続けると考えられる．

RSS Feed には，サイトの更新に関する情報が共通の書式で含まれているため，例えば，RSS リーダーと呼ばれるアプリケーションを利用することで，容易にサイトの更新情報を閲覧することが可能になり，さらには複数の RSS Feed の情報をまとめて閲覧，利用することなどが容易に実現できる．また，サイトの更新情報を配信する以外にも，メタデータ公開のためのコンテナとしての可能性も秘めており，現在様々な形で利用されている [神崎 01].

しかしながら，RSS によりメタデータを配信しているサイトは，CMS(Content Management System) を利用しているような一部のサイトに限られているのが現状である．なぜなら，CMS を使えば，ユーザは HTML 文書を生成すると同時に，自動的に RSS を生成することができるが，人手で作成している Web ページなどでは，RSS Feed も人手で作成し，メンテナンスしなければならず，これにはコストがかかるためである．また別の問題として，RSS Feed は一般的に，直近 n 回の更新に対するメタデータであるため，過去の情報を取得できないという問題がある [Karger 04].

そこで本論文では，HTML 文書中に含まれる時系列情報を自動的に発見し，RSS Feed を自動的に生成する手法を提案し，本手法に基づいたアプリケーション “なんでも RSS” について紹介する．このシステムを用いることで，RSS Feed を配信しない時系列を記述する Web ページを RSS Reader で閲覧することが可能になる．また，そのような Web ページの著者は，容易に “RSS-Autodiscovery[Pilgrim 02]” 対応の Web ページを作成することが可能になる．

連絡先: 南野 朋之，東京工業大学大学院総合理工学研究科，〒 226-8504 神奈川県横浜市緑区長津田町 4259 R2-7, Tel/Fax 045-924-5294，nanno@lr.pi.titech.ac.jp

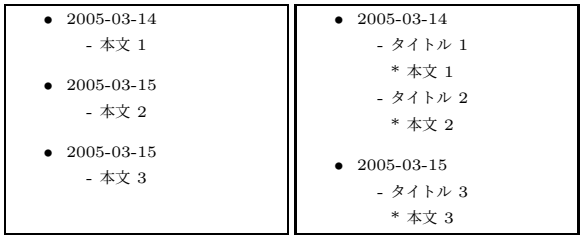

図 1: 日付ベース (左) と記事ベース (右) の時系列

2. RSS Feed 自動生成システムの概要

本節では，日付表現を伴って時系列を記述する Web ページから，RSS を生成する手法について概要を述べる．

RSS Feed は，channel 要素と item 要素の二種類の情報から構成される [RSS-DEV Working Group 00, 神崎 01]．channel 要素では，Feed のタイトル，URI，概要など，Feed 全体に関する情報, 及び，含んでいる記事の一覧などが記述される．item 要素では，channel 要素で一覧を示した各記事について，記事のタイトル，URI，概要などが記述される．

よって，Web ページ中に含まれる時系列情報から RSS Feed を自動生成するためには，適切な item 要素 (言い換えれば，“記事の範囲”) を抽出しなければならない．しかしながら，Web ページ中の時系列情報は，様々なスタイルで記述される．そこで，本研究で対象とする時系列情報は，明示的な日付表現を含む Web ページに限定する．これは，提案手法が繰り返し出現する日付表現を利用した構造解析手法に基づくためである．

図 1 に時系列情報を記述する際の典型例を示す．図 1(左) に示す日付ベースの時系列は，各記事に一つずつ日付表現が含まれる構造である．BBS や Web 日記，サイトの更新情報などを記述する時系列の多くが，このタイプに当てはまる．一方，Weblog やニュースのヘッドラインのページなどに多い，図 1(右) に示す記事ベースの時系列では，一つの日付表現に複数の記事が含まれる構造を示す．よって，このような Web ページから適切な RSS Feed を生成するためには，図 2 に示すような構造を抽出しなければならない．

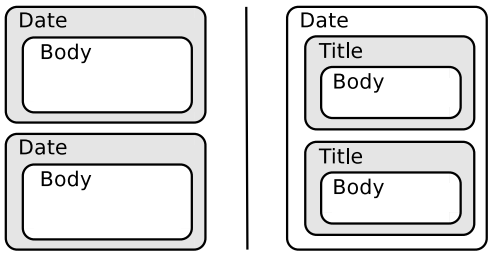

図 2: 抽出すべき構造 (日付ベースの構造 (左) と記事ベースの構造 (右))

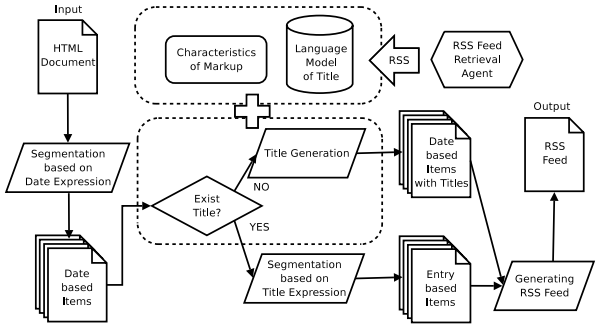

図 3: RSS Feed 自動生成システムのフローチャート

図 3 に，本論文で提案する RSS Feed 自動生成システムのフローチャートを示す．システムはまず，日付表現を検出し，3.2 節で述べる手法を適用することで，日付ベースの item を抽出する．その後，日付ベースの item がタイトル表現を含むかどうかを調べ，含まない場合は，適切なタイトルを自動生成することにより，日付ベースの RSS Feed を生成する．タイトル表現を含む場合は，さらにタイトル表現に基づいたセグメンテーションを行い，記事ベースの RSS Feed を生成する．

3. HTML 文書からの RSS Feed 自動生成

本節では，HTML 文書からの RSS Feed 自動生成手法について述べる．なお，本手法で対象とする時系列情報は，以下の性質をもつと仮定している．

1. 記事の書かれた日を示す日付表現は記事の上部にある*1
2. 一連の記事に対して，各日付表現に係るタグの種類は一定である
3. 一連の日付表現はフォーマットが一定である
   ex.) “2003/1/2” と “2-Jan-2003” は別のフォーマット

以下では，図 3 に従い，それぞれの処理について詳説する．

3.1 日付表現の抽出

システムはまず，HTML Tidy[Raggett 94] により，well-formed な XML 文書に変換した HTML 文書から，日付表現を抽出する．日付表現の抽出には，人手により作成した正規表現ベースのルールを使用する．最上部に年が記載され，各記事の日付では年を省略し，月日のみを記載するようなケースや，年号の省略，西暦の上二桁の省略に対応するために，ヒューリスティックを用い，不足している情報の補完も行う．(日付表現の検出に関する詳細については，[南野 04] を参照．)

*1 フラットな構造でマークアップされる場合のみ，この性質を仮定する．マークアップにより日付と記事の対応が明らかである場合 (例えば，日付表現と記事を囲うようなタグが存在する場合) は，この限りではない．

3.2 日付ベースの記事の抽出

次に，システムは検出された日付表現を使用し，Web ページを日付ベースの記事に分割する．

Web ページには，複数の時系列情報が記述される可能性がある．そこで，システムはまず前節で発見された日付表現を “同じタイプの日付表現” 毎にグルーピングする．“同じタイプの日付表現” とは以下の条件をすべて満たす日付表現の集合を指す．

- タグの係り方が同じ
- 日付表現を囲む HTML タグ中において，タグの直後 (もしくは直前) から日付表現の直前 (もしくは直後) までの byte 数が同じ
  ex.) `<h1>` ☆ 2005.2.2　発表申込締切 `</h1>`
  `<h1>` ☆ 2005.4.8　論文投稿のページを追加 `</h1>`
  以上二つの日付表現は，タグの直後から日付表現の直前までの byte 数が同じ (どちらも 2byte) である．
- 日付表現のフォーマットが同じ
  ex.) “2004/1/1” と “1/2” は同じフォーマット
  “1-Jan-2004” と “2004/1/2” は別フォーマット

これら同じタイプの日付表現は，ブラウザによってレンダリングされた際に同じように表示されるため，これらを一連の日付表現と考え，システムは各タイプに対して，Web ページから日付ベースの記事を抽出することを試みる．

システムはまず，あるタイプに属する日付表現に対し，各日付表現によって修飾される記事の開始位置を，以下の手順で決定する．

1. 同じタイプに属する日付表現に対し，XPath[Clark 99] 表現を列挙する
   (同じタイプに属する日付表現は，タグの係り方は同一である)
2. それらの表現を根ノードから順に見ていき，すべての日付表現に対して共通でない最初のタグを記事の開始位置とする

図 4 の例を考える．この文書には，三つの日付表現 (“date 1”，“date 2”，“date 3”) が存在し，それらは同じタイプに属しているとする．これらの日付表現に対する XPath 表現は以下の通りである．

- date 1:　/body[1]/div[1]/h1[1]/date[1]
- date 2:　/body[1]/div[1]/h1[2]/date[1]
- date 3:　/body[1]/div[3]/h1[1]/date[1]

“/body[1]/div[3]” は，HTML 文書中の一つ目の body タグで囲まれた部分に含まれる div タグのうち，前から三番目の div タグで囲まれた範囲を示す．

これら三つの XPath 表現を見ていくと，“div” の直下にある “h1” で初めてすべての日付表現に対して別のタグが修飾している．(“body” は，三つの日付表現で共通．“div” は “date 1” と “date 2” で共通．) これは，深さ 3 の位置にある h1 タグでセグメンテーションを行えば，各セグメントにちょうど一つの日付表現が属することを示している．つまり，この深さ 3

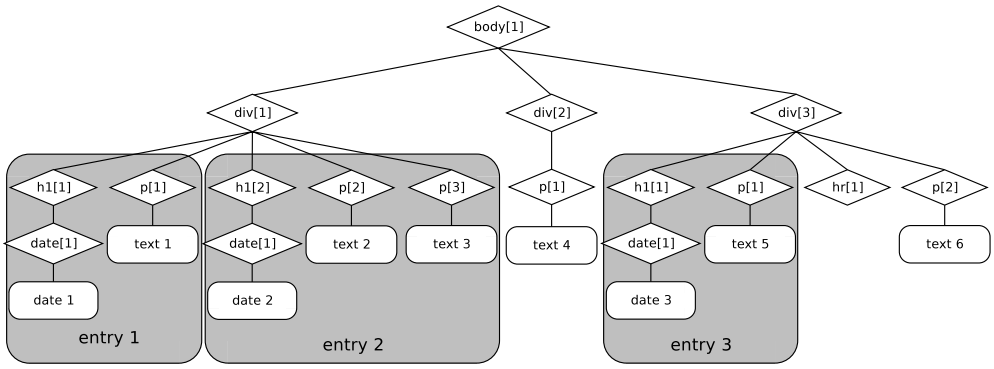

図 4: 日付ベースの記事の抽出例

の位置にある h1 タグは記事の開始位置を示すと考えることが出来る．

次にシステムは，記事の終了位置を決定する．システムはまず，以下のルールによって記事の終了位置の候補を抽出する．

(1) 次の記事の開始位置の一つ前の位置

(2) DOM ツリーにおいて，開始位置のタグよりも根ノードに近い HTML タグが表れる一つ前の位置

ただし，最後の記事については，上記 (1) の条件が使用できないため，以下の条件を追加する．

(3) 開始位置のタグと DOM 中の深さが同じタグのうち，それ以前の entry で一度も出現していないタグが現れる一つ前の位置

次に，これらの候補位置の中からもっとも開始位置に近い終了位置を最終的な終了位置として選択する．

先ほど同様，例として図 4 を考える．図中の “date 1” に対する記事は，“text 1” を含む．なぜなら，この記事の終了位置はルール (1) によって決定されるためである．(二つ目の記事の開始位置は “date 2” を修飾する “h1” タグである．) “date 2” が示す二つ目の記事は，“text 2” と “text 3” を含むが，“text4” は含まない．なぜなら，この記事の終了位置はルール (2) によって決定されるためである．(“text 4” を修飾する “div” タグは開始位置の “h1” タグよりも根ノードに近い．) また，“date 3” が示す最後の記事は，“text 5” を含むが “text 6” は含まない．なぜなら，この記事の終了位置はルール (3) によって決定されるためである．(“hr” タグは，この記事より前に表れるどの記事にも含まれていない．)

以上のような処理により各日付表現が修飾する記事の開始位置と終了位置を決定することで，それぞれの日付表現のタイプに対して日付ベースの記事集合を決定することができる．

3.3 タイトルの検出

次にシステムは，それぞれの日付ベースの記事がタイトル表現を含むかどうかを判定する．タイトルの検出は，タイトルをマークアップしているタグを発見するという問題と考え，以下のような特徴を利用し行う．

1. すべての日付単位の記事に最低一度以上出現しなければならない
2. そのタグでマークアップされている範囲に，タイトルとして適切な長さの text が存在しなければならない
3. そのタグでマークアップされている範囲に，改行効果のあるタグがあってはならない
4. 日付表現と，そのタグの中で最初に現れるタグ間に出現する text が閾値以下でなければならない (日付と最初のタイトルとの間に，長い text があるのは不適当．)

また，以下のような特徴をもつタグは，タイトルをマークアップしやすいと考え，優先的に選択を行う．

1, heading タグ (`h1..h6`) である

2, class 名に “title” や “head” などを含む

3, タグがマークアップするすべての text について
- 平均文字長が 20 文字以内
- RSS Feed より収集した約 300 万のタイトルから構築した品詞の 3-gram モデルに適合

タイトル検出の詳細については，[南野 05] を参照．

3.4 日付ベースの RSS Feed の生成

3.3 節で述べたタイトル検出処理により，タイトルが含まれていないと判定された場合，日付ベースの RSS Feed を生成する．

その際，タイトルの生成を行う必要がある．なぜなら，RSS の item 要素のタイトルは，必須項目であるだけでなく，ユーザが本文を読むかどうかの重要な指標となるためである．

タイトルの生成には extraction ベースの手法を用い，“タイトル候補の抽出”，“適切なタイトルの選択” の二段階手法で行う．タイトルの生成は，`ping.bloggers.jp`[*2]で，公開されている情報を元に大量に収集した，主に Weblog の RSS Feed から得られる統計情報を元に行う．

“タイトル候補の抽出” の部分では，約 300 万のタイトルから生成した品詞の 3-gram に基づくタイトルの言語モデルを使用し，記事本文からタイトルの候補を生成する．“適切なタイトルの選択” の部分では，式 1 に示す式でランキングし，最もスコアの高い候補をタイトルとして選択する．

$$score(w_1, ..., w_n) = P_{length}(n) \quad (1)$$
$$* \quad \sum_{i=1}^{n} tf * idf(w_i)^2 * P_{Date}(w_i \in T_{Date} | w_i \in D_{Date})$$

なお，式 1 の右辺の第一項は，長さ n のタイトルが生成される確率，第二項は，ある時期にその単語が本文に現れた際，そ

*2 `http://ping.bloggers.jp/`

図 5: JSAI2005 サイトのスクリーンショット

図 6: なんでも RSS の出力結果

の単語がタイトルに含まれる確率を示している．なお，これらの確率も RSS Feed から得られる本文とタイトルのペアを継続的に収集することにより計算されている．

タイトル生成の詳細については，[南野 05] を参照．

3.5 記事ベースの RSS Feed の生成

3.3 節で述べたタイトル検出処理により，タイトルが含まれていると判定された場合，さらに構造解析を行い，記事ベースの RSS Feed を生成する．

再度，図 2 を考える．図 2 に示す，日付ベースと記事ベースの構造を比較すると，日付ベースの構造における日付表現が，記事ベースの構造におけるタイトル表現と対応していることがわかる．言い換えれば，3.2 節で述べた，日付表現に対して適用した構造解析手法を，日付表現により分割した各記事に含まれるタイトル表現に対して，再び適用することで，記事ベースの構造を抽出することが可能である．

以上のような手続きにより，記事ベースの RSS Feed を生成する．

4. なんでも RSS

本節では，本論文で提案した手法を使用したアプリケーション，“なんでも RSS[*3]” について述べる．

図 5 は，JSAI2005[*4] のスクリーンショットの一部である．このサイトには，“最新情報” セクションには，サイトの変更箇所が日付表現を伴って記述されている．図 6 は，“なんでも RSS” が出力した RSS Feed の一つである．

*3 http://blogwatcher.pi.titech.ac.jp/nandemorss/
*4 http://www.jaist.ac.jp/jsai2005/

このように，本システムは，Web ページ中のタイトル表現を発見し，それぞれの日付表現が修飾する適切な Web ページの一部を切り出し，RSS Feed を生成する．また，Web ページが複数の時系列を含む場合は，すべての可能性のある Feed を作成し，ユーザはそれらの中から自分の欲しい Feed を選択することが可能である．

このシステムと RSS Reader のようなアプリケーションを組み合わせることで，ユーザは，このサイトの変更を容易に追跡することが可能になる．また，別の利用方法としては，このシステムが出力する RSS Feed の URL を，この Web ページの header に指定することで，追加のメンテナンスコスト無しで，“RSS-Autodiscovery” 対応の Web ページを作成することが可能になる．

5. おわりに

本論文では，時系列情報を記述する Web ページ中の日付表現とタイトルを検出することにより，RSS Feed を自動的に生成する手法を提案した．

RSS によるメタデータ配信が盛んに行われるようになってきたが，RSS Feed を出力するサイトは，CMS を使う一部のサイトに限られているのが現状である．なぜなら，人手で記述されるような Web サイトでは，追加のメンテナンスコストが必要になるためである．

今後の課題は，提案手法の評価である．評価は，(1) 構造解析に関する評価，(2) タイトル生成に関する評価の二点を行う予定である．また，さらなる課題として，生成された Feed のタイプ (掲示板，weblog，新着情報，イベント案内など) の自動的な判定や，取得した item の内容に基づく分類などが考えられる．

なお，本手法に基づいて構築したシステム “なんでも RSS” は，http://blogwatcher.pi.titech.ac.jp/nandemorss/ で公開中である．

参考文献

[Clark 99] Clark, J. and DeRose, S.: XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath (1999)

[Karger 04] Karger, D. R. and Quan, D.: What Would It Mean to Blog on the Semantic Web?, in *Proc. Third International Semantic Web Conference*, pp. 214–228 (2004)

[Libby 99] Libby, D.: RSS 0.91 Spec, revision 3, http://my.netscape.com/publish/formats/rss-spec-0.91.html (1999)

[Pilgrim 02] Pilgrim, M.: dive into mark - RSS autodiscovery, http://diveintomark.org/archives/2002/05/30/rss_autodiscovery (2002)

[Raggett 94] Raggett, D.: Clean up your Web pages with HTML TIDY, http://www.w3.org/People/Raggett/tidy/ (1994)

[RSS at Harvard Law 02] RSS at Harvard Law, : RSS 2.0 Specification, http://blogs.law.harvard.edu/tech/rss (2002)

[RSS-DEV Working Group 00] RSS-DEV Working Group, : RDF Site Summary (RSS) 1.0, http://web.resource.org/rss/1.0/spec (2000)

[神崎 01] 神崎 正英：RSS – サイト情報の要約と公開, http://www.kanzaki.com/docs/sw/rss.html (2001)

[南野 04] 南野 朋之, 鈴木 泰裕, 藤木 稔明, 奥村 学：blog の自動収集と監視, 人工知能学会論文誌, Vol. 19, No. 6, pp. 511–520 (2004)

[南野 05] 南野 朋之, 奥村 学：RSS 自動生成のためのタイトル生成, 言語処理学会第 11 回年次大会, pp. 57–60 (2005)